NASA/CR-2001-211042
ICASE Report No. 2001-22

Robust Airfoil Optimization to Achieve Consistent Drag Reduction Over a Mach Range

Wu Li
Old Dominion University, Norfolk, Virginia

Luc Huyse
ICASE, Hampton, Virginia

Sharon Padula
NASA Langley Research Center, Hampton, Virginia

ICASE
NASA Langley Research Center
Hampton, Virginia
Operated by Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-97046

August 2001

ROBUST AIRFOIL OPTIMIZATION TO ACHIEVE CONSISTENT DRAG REDUCTION OVER A MACH RANGE *

WU LI†, LUC HUYSE‡, AND SHARON PADULA§

Abstract. We prove mathematically that in order to avoid point-optimization at the sampled design points for multipoint airfoil optimization, the number of design points must be greater than the number of free-design variables. To overcome point-optimization at the sampled design points, a robust airfoil optimization method (called the profile optimization method) is developed and analyzed. This optimization method aims at a consistent drag reduction over a given Mach range and has three advantages: (a) it prevents severe degradation in the off-design performance by using a smart descent direction in each optimization iteration, (b) there is no random airfoil shape distortion for any iterate it generates, and (c) it allows a designer to make a trade-off between a truly optimized airfoil and the amount of computing time consumed. For illustration purposes, we use the profile optimization method to solve a lift-constrained drag minimization problem for 2-D airfoil in Euler flow with 20 free-design variables. A comparison with other airfoil optimization methods is also included.

Key words. robust optimization, airfoil shape optimization, consistent drag reduction, lift-constrained drag minimization, adaptive weight adjustment

Subject classification. Applied and Numerical Mathematics

1. Introduction. Optimization of aerospace vehicles or aircraft wings is based on a mathematical model of the physical reality. The design variables are parameters in the mathematical model and changes in these design variables result in new physical objects. Aerodynamic optimization is a process of finding a set of design variables that corresponds to a new physical object with better aerodynamic and structural properties.

For airfoil shape optimization, design variables are parameters that define the airfoil shape. One accepted practice for modeling airfoil shapes is to use empirical algebraic expressions that are based on the knowledge of aerodynamic properties of airfoils. There are two advantages to such an airfoil model:

(a) Each set of design variables (such as maximum thickness, camber, radius of leading edge, etc.) generates an airfoil with aerodynamic properties that are well understood by experts.

(b) The number of design variables is small, thus the corresponding optimization problem is computationally affordable.

A drawback to such an airfoil model is that the optimization process is merely selecting a desirable airfoil shape among the predetermined shapes given by the specific airfoil model. The usefulness of the optimal airfoil is essentially determined by the usefulness of the airfoil model.

To achieve truly innovative airfoil designs, it is therefore desirable to consider "free-form" shape optimization, which allows "truly optimum" designs to be computed [5, Section 1]. Here "free-form" means geometric shapes represented by a linear combination of general basis functions (such as splines or sinusoidal

*This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 while the first two authors were in residence at ICASE, NASA Langley Research Center, Hampton, VA 23681.

†Department of Mathematics and Statistics, Old Dominion University, Norfolk, VA 23529 (email: wli@odu.edu).

‡ICASE, NASA Langley Research Center, Hampton, VA 23681 (email: luc@icase.edu).

§MDO Branch, NASA Langley Research Center, Hampton, VA 23681 (email: S.L.PADULA@LaRC.NASA.GOV).

basis functions). However, there are several challenges associated with "free-form" airfoil shape optimization. Drela [5, Subsection 5.1] pointed out that if presented with sufficient design model resolution, an optimizer will readily (and annoyingly) manipulate and exploit the flow at the smallest significant physical scales present that tends to produce improved performance only near the sampled operating conditions. The point-optimized airfoil often shows a possibly severe degradation in the off-design performance and optimized aerodynamic shapes are usually "noisy" and usually require a posteriori smoothing.

The main objective of this paper is to develop a robust airfoil optimization scheme for achieving consistent drag reduction over a given Mach range. This scheme (called the profile optimization method) has the following three advantages: (a) it prevents severe degradation in the off-design performance by using a smart descent direction in each optimization iteration, (b) there is no random airfoil shape distortion for any iterate it generates, and (c) it allows a designer to make a trade-off between a truly optimized airfoil and the amount of computing time consumed.

The term "robustness" has been used to mean a variety of things. For optimization under uncertainty, we can distinguish the following meanings and goals of "robust optimization":

1. Identify designs that minimize the variability of the performance under uncertain operating conditions. This is the objective of Taguchi methods [8], which are most practical when the expected value of the performance can be adjusted at negligible cost.
2. Mitigate the detrimental effects of the worst-case performance. This is the objective of minimax strategies, which can be used to find a design with the optimal worst-case performance [4].
3. Provide the best overall performance of a system by maximizing the expected value of its utility [12].
4. Achieve consistent improvements of the performance over a given range of uncertainty parameters. This is the main objective of this work.

The paper is organized as follows. In Section 2, we give a brief introduction to the airfoil optimization problem. Section 3 is devoted to Drela's hypothesis on the necessity of using at least $\mathcal{O}(m)$ design points for multipoint airfoil shape optimization, where m is the number of free-design variables. The main result is a mathematical proof of the fact that the multipoint airfoil shape optimization needs at least $(m+1)$ design points. We present two robust airfoil shape optimization formulations in Section 4 and prove the mathematical equivalence between these two formulations under finite sampling in Section 5. The profile optimization method is introduced in Section 6. The implementation issues of the profile optimization method are discussed in Section 7 and numerical simulation results are given in Section 8. Final conclusions are drawn in Section 9.

2. Airfoil Shape Optimization. Recently, there has been significant progress in airfoil shape optimization (see [2, 3, 5, 16] and references therein). These papers demonstrate impressive shape optimization using high-fidelity CFD codes, reliable grid generation, and numerically efficient sensitivity calculations. Equally impressive progress has been made in optimization of 3-D wings [6, 7, 17] and in coupled structural-aerodynamic optimization [9]. However, except Drela's work [5], these aerodynamic shape optimization projects all find optimal shapes based on fixed operating conditions. In this paper, we study a simplified 2-D airfoil shape optimization problem using a low-fidelity Euler flow solver, but we include uncertainty in the operating conditions for the airfoil shape optimization.

Airfoil shape optimization is a PDE-constrained optimization problem. A general mathematical framework for airfoil optimization can be described as follows (see [3, 16, 17] for details).

For a given flow and turbulence model (a set of PDEs), one could compute the energy, velocity, and

pressure around a given airfoil, which will be denoted by one vector-valued function $\hat{S}$. For an airfoil optimization problem (which is a 2-D problem), $\hat{S}$ is a function of (x, y) for fixed (D, α, M), where D denotes the set of geometric design parameters that defines the airfoil, α is the angle of attack, and M is a given Mach number. Note that M and α are parameters representing the flight conditions. The PDE equations for $\hat{S}$ are solved by a numerical method that involves grid generation, discretization of the PDEs, and iterations for solving systems of nonlinear equations.

For a given geometric design vector D, the generated grid (denoted by X) is determined by D, *i.e.*, $X = X(D)$ is a function of D. Usually, $X(D^0)$ is generated for an initial design vector D^0 based on a grid generation method, and $X(D)$ (for $D \neq D^0$) is generated by a grid movement strategy (such as the spring analogy or the elasticity model) [16].

The system of nonlinear equations derived from a discretization of the PDEs for the flow and turbulence model will be denoted by

$$R(D, \alpha, M, X, S) = 0, \tag{2.1}$$

where S is a vector whose components are values of $\hat{S}$ at the grid points (or the cell centers). For any given (D, α, M, X), a numerical solution of (2.1) gives S.

If we consider minimization of the drag coefficient under a minimal requirement on the lift coefficient, then the multipoint airfoil optimization problem can be formulated as follows [5]:

$$\min_{D,\alpha_i} \sum_{i=1}^{r} w_i \, c_d\Big(D, \alpha_i, M_i, X(D), S_i(D, \alpha_i)\Big) \tag{2.2}$$

subject to

$$c_l\Big(D, \alpha_i, M_i, X(D), S_i(D, \alpha_i)\Big) \geq c_l^* \quad \text{for } 1 \leq i \leq r \tag{2.3}$$

and $D \in \mathcal{F}$, where w_i's are positive weights, c_l^* is the minimal lift value, $\mathcal{F}$ is a given feasible set for geometric design variables, $S_i(D, \alpha_i)$ is the solution of (2.1) for $M = M_i$ and $\alpha = \alpha_i$, and c_d (or c_l) denotes an approximate value of the drag (or lift) coefficient. The feasible set $\mathcal{F}$ mainly depends on the airfoil model. In this paper, we use splines for airfoil shape modeling: the components of D are the control points for a spline curve and the feasible set $\mathcal{F} = \mathbb{R}^m$, where m is the number of geometric design variables.

Recently, Drela [5] studied the behavior of the optimization solutions of (2.2) in two-dimensional viscous flow when the number of free-design variables is relatively large. Drela [5] concluded that increasing the number of geometric design variables requires a corresponding increase in the number of design conditions (Mach numbers) used in the multipoint optimization problem (2.2). He also suggested that it is necessary to have $r = \mathcal{O}(m)$, where m is the number of free-design variables, to achieve a smooth airfoil geometry. Other notable conclusions made by Drela [5, Conclusion] are:

- Near-continuous sampling of the operating space (*i.e.*, in the range of Mach numbers) may be required in the theoretical limit of a general airfoil design problem with a very large number of degree of freedom (for geometric variables) — a very expensive proposition.
- The most suitable operating points to be actually sampled in multipoint airfoil optimization (*i.e.*, $M_1, M_2, \ldots, M_r$) are not apparent a priori. From limited experience, sampling somewhat beyond the expected operating range appears to be best.
- The point weights (*i.e.*, $w_1, w_2, \ldots, w_r$) used in multipoint airfoil optimization are arbitrary, and their appropriate values can not be easily estimated without prior experience.
- Optimized aerodynamic shapes are usually "noisy" and require a posteriori smoothing.

3. The Critical Number of Design Points for Multipoint Airfoil Optimization. In this section, we give a mathematical proof of Drela's hypothesis on the necessity of having the number of design points proportional to the number of design variables. Specifically, we prove that in order to avoid point-optimization at the design points, the number of design points must be at least $(m+1)$, where m is the number of free-design variables.

In (2.2), the angle of attack, α_i, is predominantly determined by the constraint on the lift coefficient at M_i. Therefore, it is theoretically possible to eliminate the constraint and consider the following unconstrained reformulation of (2.2):

$$\min_D \sum_{i=1}^{r} w_i f(D, M_i), \tag{3.1}$$

where M_i's are design points, w_i's are positive weights, and f is a function related to the drag coefficient.

Let $\hat{D}$ be an optimal solution of (3.1). Then

$$\sum_{i=1}^{r} w_i \frac{\partial f}{\partial D}(\hat{D}, M_i) = 0, \tag{3.2}$$

where $\frac{\partial f}{\partial D}$ denotes the gradient of f with respect to D.

If $r \leq m$, then $(r-1) < m$. So we can find a nonzero vector ΔD in the orthogonal complement of the subspace of $\mathbb{R}^m$ generated by the following $(r-1)$ vectors:

$$\frac{\partial f}{\partial D}(\hat{D}, M_2), \ldots, \frac{\partial f}{\partial D}(\hat{D}, M_r).$$

By (3.2) and $w_1 > 0$, ΔD must also be orthogonal to $\frac{\partial f}{\partial D}(\hat{D}, M_1)$. Therefore, we obtain a nonzero vector ΔD such that

$$\left\langle \frac{\partial f}{\partial D}(\hat{D}, M_i), \Delta D \right\rangle = 0 \quad \text{for } 1 \leq i \leq r, \tag{3.3}$$

where $\langle u, v \rangle$ denotes the dot product of vectors u and v.

Let M_{r+1} be any Mach number such that

$$\left\langle \frac{\partial f}{\partial D}(\hat{D}, M_{r+1}), \Delta D \right\rangle \neq 0. \tag{3.4}$$

Let $\{\hat{w}_1, \ldots, \hat{w}_{r+1}\}$ be a set of arbitrary positive weights. By Taylor expansion, we get

$$\sum_{i=1}^{r+1} \hat{w}_i f(\hat{D} + t\Delta D, M_i) = \sum_{i=1}^{r+1} \hat{w}_i f(\hat{D}, M_i) + t \sum_{i=1}^{r+1} \hat{w}_i \left\langle \frac{\partial f}{\partial D}(\hat{D}, M_i), \Delta D \right\rangle + \mathcal{O}(t^2). \tag{3.5}$$

It follows from (3.5) and (3.3) that

$$\sum_{i=1}^{r+1} \hat{w}_i f(\hat{D} + t\Delta D, M_i) = \sum_{i=1}^{r+1} \hat{w}_i f(\hat{D}, M_i) + t\hat{w}_{r+1} \left\langle \frac{\partial f}{\partial D}(\hat{D}, M_{r+1}), \Delta D \right\rangle + \mathcal{O}(t^2). \tag{3.6}$$

By (3.4), we can choose t such that $|t|$ is sufficiently small and

$$t \left\langle \frac{\partial f}{\partial D}(\hat{D}, M_{r+1}), \Delta D \right\rangle < 0.$$

Then (3.6) implies

$$\sum_{i=1}^{r+1} \hat{w}_i f(\hat{D} + t\Delta D, M_i) < \sum_{i=1}^{r+1} \hat{w}_i f(\hat{D}, M_i).$$

That is, after adding one more design point, $\hat{D}$ is not an optimal solution no matter how the weights are selected. ■

In summary, if the number of design points is less than the number of free-design variables, then it is possible to have a first-order reduction of the value of f at some Mach number M_{r+1} (in order of $|t|$) and only second-order increments of the values of f at M_i's ($1 \le i \le r$) (in order of t^2). This results in a much better performance of the airfoil at an off-design point M_{r+1} with a marginal deterioration of the performance at other Mach numbers.

4. Robust Airfoil Optimization Formulations. For engineering design problems, an optimal design is usually obtained under some explicit/implicit assumptions. This leads to a design that works well under ideal operating conditions but may perform inadequately under non-ideal (*i.e.*, off-design) conditions. The problem is that the optimal design does not consider the uncertainty or variability of some parameters/data that will affect the actual performance of the design in a real-world situation. Therefore, it is necessary to include uncertainties in a practical design optimization process. In this paper, we assume that M is the only uncertain parameter and $[M_{\min}, M_{\max}]$ is the range of Mach numbers considered.

Let $p(M)$ be the probability density function of the uncertain parameter M. Then a stochastic programming formulation for airfoil optimization under uncertainty can be described as follows [12]:

$$\min_{D,\alpha(\cdot)} \int_{M_{\min}}^{M_{\max}} c_d(D, M, \alpha(M))\, p(M)\; dM \tag{4.1}$$

subject to

$$c_l(D, M, \alpha(M)) \ge c_l^* \quad \text{for } M_{\min} \le M \le M_{\max} \tag{4.2}$$

and $D \in \mathcal{F}$, where $\mathcal{F}$ is the feasible geometric design space and c_l^* is the minimal lift value. This corresponds to the third definition of "robust optimization" given in the introduction.

On the other hand, one can also use the following minimax optimization formulation for robust optimization under uncertainty [4]:

$$\min_{D,\alpha(\cdot)} \max_{M_{\min} \le M \le M_{\max}} \rho(M)\, c_d(D, M, \alpha(M)) \tag{4.3}$$

subject to the constraints given in (4.2). Here $\rho(M) > 0$ is a positive weighting function of M. This corresponds to the second definition of "robust optimization" given in the introduction.

The constraints on the lift can be eliminated by choosing an appropriate value for the angle of attack corresponding to each M. In fact, Elliott and Peraire [6] suggested to incorporate a means for adjusting the angle of attack to satisfy the lift constraint into the flow analysis algorithm. One problem with this elimination approach is that the angle of attack becomes a function of M and D, and the derivatives of α with respect to D have to be computed in order to get the derivatives of c_l and c_d with respect to D. In this study, we retain the lift constraint to make our methods applicable to general constrained aerodynamic optimization problems (with constraints on the pitching moment and the leading edge radius, etc.).

5. Approximations to Robust Optimization Formulations. Since we cannot compute c_l and c_d for all M in the Mach range $[M_{\min}, M_{\max}]$, computationally tractable approximations of (4.1) and (4.3) must be used. The simplest approximation scheme is to replace $[M_{\min}, M_{\max}]$ by a finite subset of $[M_{\min}, M_{\max}]$, say $\{M_1, M_2, \ldots, M_r\}$. Then (4.1) is reduced to the following multipoint optimization problem [5]:

$$\min_{D,\alpha_1,\ldots,\alpha_r} \sum_{i=1}^{r} w_i\, c_d(D, M_i, \alpha_i) \tag{5.1}$$

subject to

$$D \in \mathcal{F}, \quad c_l(D, M_i, \alpha_i) \geq c_l^* \quad \text{for } 1 \leq i \leq r, \tag{5.2}$$

where the weights w_i depend on the probability density function $p(M)$ and a chosen integration scheme. Similarly, the minimax formulation (4.3) can be discretized as follows:

$$\min_{D,\alpha_1,\ldots,\alpha_r} \max_{1\leq i\leq r} \rho_i\, c_d(D, M_i, \alpha_i), \tag{5.3}$$

subject to the constraints given in (5.2). Here $\rho_i > 0$ is determined by $\rho(M)$ and M_i.

It seems that (5.1) and (5.3) are completely different. However, under the strict complementarity condition (*i.e.*, Lagrange multipliers are nonzero for all active constraints), (5.1) is mathematically equivalent to (5.3). Here we give a proof of the mathematical equivalence between (5.1) and (5.3).

Let $(\hat{D}, \hat{\alpha}_1, \ldots, \hat{\alpha}_r)$ be a stationary solution of (5.1). For simplicity, assume that $\hat{D}$ is in the interior of $\mathcal{F}$ and the equality holds for lift constraints. Then, under the strict complementarity condition, the following first-order optimality condition (or the KKT-condition) for (5.1) holds:

$$\begin{aligned}
&\sum_{i=1}^{r} w_i \frac{\partial c_d}{\partial D}(\hat{D}, M_i, \hat{\alpha}_i) - \sum_{i=1}^{r} \lambda_i \frac{\partial c_l}{\partial D}(\hat{D}, M_i, \hat{\alpha}_i) = 0, \\
&w_i \frac{\partial c_d}{\partial \alpha_i}(\hat{D}, M_i, \hat{\alpha}_i) = \lambda_i \frac{\partial c_l}{\partial \alpha_i}(\hat{D}, M_i, \hat{\alpha}_i) \text{ for } 1 \leq i \leq r, \\
&\lambda_i > 0,\ c_l(\hat{D}, M_i, \hat{\alpha}_i) = c_l^* \quad \text{for } 1 \leq i \leq r.
\end{aligned} \tag{5.4}$$

Define

$$\begin{aligned}
\hat{\gamma} &:= \sum_{i=1}^{r} w_i c_d(\hat{D}, M_i, \hat{\alpha}_i), \\
\rho_i &:= \frac{\hat{\gamma}}{c_d(\hat{D}, M_i, \hat{\alpha}_i)} > 0 \quad \text{for } 1 \leq i \leq r, \\
\theta_i &:= \frac{w_i}{\rho_i} \quad \text{for } 1 \leq i \leq r.
\end{aligned}$$

Then (5.4) implies the following conditions:

$$\begin{aligned}
&\sum_{i=1}^{r} \theta_i \rho_i \frac{\partial c_d}{\partial D}(\hat{D}, M_i, \hat{\alpha}_i) - \sum_{i=1}^{r} \lambda_i \frac{\partial c_l}{\partial D}(\hat{D}, M_i, \hat{\alpha}_i) = 0, \\
&\theta_i \rho_i \frac{\partial c_d}{\partial \alpha_i}(\hat{D}, M_i, \hat{\alpha}_i) = \lambda_i \frac{\partial c_l}{\partial \alpha_i}(\hat{D}, M_i, \hat{\alpha}_i) \text{ for } 1 \leq i \leq r, \\
&\lambda_i > 0,\ c_l(\hat{D}, M_i, \hat{\alpha}_i) = c_l^* \quad \text{for } 1 \leq i \leq r,
\end{aligned} \tag{5.5}$$

$$\rho_i c_d(\hat{D}, M_i, \hat{\alpha}_i) = \hat{\gamma} \quad \text{for } 1 \le i \le r,$$

$$\theta_i > 0 \quad \text{for } 1 \le i \le r, \text{ and } \sum_{i=1}^{r} \theta_i = 1.$$

By (5.5), $(\hat{\gamma}, \hat{D}, \hat{\alpha}_1, \ldots, \hat{\alpha}_r)$ satisfies the KKT-condition of the following optimization problem:

$$\min_{D, \alpha_i, \gamma} \ \gamma \quad \text{subject to} \quad c_l(D, \alpha_i, M_i) \ge c_l^* \ \text{ and } \ c_d(D, \alpha_i, M_i) \le \frac{\gamma}{\rho_i} \quad \text{for } 1 \le i \le r. \tag{5.6}$$

Since (5.6) is mathematically equivalent to (5.3), $(\hat{D}, \hat{\alpha}_1, \ldots, \hat{\alpha}_r)$ is also a stationary point of (5.3).

On the other hand, assume that $(\hat{D}, \hat{\alpha}_1, \ldots, \hat{\alpha}_r)$ is a stationary point of (5.3), the equality holds in (5.2), and $\rho_i c_d(\hat{D}, M_i, \hat{\alpha}_i) = \hat{\gamma}$ for $1 \le i \le r$, where

$$\hat{\gamma} := \max_{1 \le i \le r} c_d(\hat{D}, M_i, \hat{\alpha}_i).$$

Then $(\hat{\gamma}, \hat{D}, \hat{\alpha}_1, \ldots, \hat{\alpha}_r)$ is a stationary solution of (5.6). If we further assume that the strict complementarity condition holds, then the KKT-condition (5.5) for (5.6) holds. It follows from (5.5) that (5.4) holds for $w_i = \rho_i \theta_i$. Therefore, $(\hat{D}, \hat{\alpha}_1, \ldots, \hat{\alpha}_r)$ is also a stationary point of (5.1). ■

REMARK 1. Note that the strict complementarity condition holds naturally for (5.1). In fact, if the lift is greater than its target value, then one can reduce the angle of attack so that the lift with respect to the new angle of attack is equal to the target value, but the drag with respect to the new angle of attack is reduced. This is not possible at a stationary point. Therefore, all lift constraints at a stationary point become equality constraints. Also, the derivative of the drag with respect to the angle of attack is positive for the range of the angle of attack considered herein, which implies that the Lagrange multipliers λ_i are positive (cf. (5.4) or (5.5)). Thus, the strict complementarity condition for (5.1) always holds. An implication is that it is possible to recover an optimal solution of (5.1) by solving (5.3) with appropriate choices of ρ_i's.

However, it is a little more difficult to claim that a stationary point of (5.3) is also a stationary point of (5.1). In the proof given above, we used two additional assumptions: (a) all $\rho_i c_d(\hat{D}, M_i, \hat{\alpha}_i)$ have the same value $\hat{\gamma}$, and (b) the strict complementarity condition holds for the drag constraints in (5.6). The first assumption is not realistic in the sense that we do not know how to choose ρ_i's to make $\rho_i c_d(\hat{D}, M_i, \hat{\alpha}_i)$ have the same value for an optimal solution. Adaptive adjustments of ρ_i's are used in the profile optimization method to ensure all $\rho_i c_d(D, M_i, \alpha_i)$ $(1 \le i \le r)$ have the same value during the optimization iterations. The assumption (b) is a natural one, since the strict complementarity condition for the drag constraints means that every drag constraint is necessary for getting the optimal solution. ■

The results in [5] indicate that the multipoint airfoil optimization formulation (5.1) tends to produce airfoils that have possibly severe degradation in the off-design performance, which Drela referred to as point-optimization behavior. The above equivalence analysis also implies that the same conclusion holds for the minimax formulation (5.3).

Huyse and Lewis [12] concluded that the point-optimization behavior can be attributed to the discretization error between (4.1) and (5.1). The multipoint formulation (5.1) approximates the integral as a discrete sum: low drag is obtained at the selected Mach numbers $M_1, \ldots, M_r$, but there is no control over, nor requirements for, the drag at the other Mach numbers. Due to the highly nonlinear nature of the PDEs for flow simulation, the optimizer is able to mold the objective function c_d to its own advantage: distinct drag troughs appear at each of the Mach numbers $M_1, \ldots, M_r$ (see also [5]). To prevent the optimization algorithm from exploiting the approximation error, Huyse and Lewis [12] suggest changing the integration points $M_1, \ldots, M_r$ during each optimization step. This effectively ensures that low drag is obtained not

for just r fixed Mach numbers, but for any combination of r Mach numbers $M_1, \ldots, M_r$. In their study, they added a random perturbation to the integration points in each optimization step but state that "any adaptive scheme that varies the location of the integration points M_k for each optimization step will do."

We will show that with only far fewer design points than the number of free-design variables, the profile optimization method can adaptively adjust ρ_i's in (5.3) to achieve a consistent drag reduction over the given Mach range, so there will be no degradation in the off-design performance of an optimal airfoil.

6. Profile Optimization Method. In this section, we first give a motivation for a new shape optimization strategy based on the robust optimization model (5.3) and then present the profile optimization method.

Let $\hat{D}$ be a fixed feasible design vector. We can plot the drag with respect to the Mach numbers over the interval $[M_{\min}, M_{\max}]$ while keeping the lift at a constant value by adjusting the angles of attack for different Mach numbers. Such a plot will be called a profile for $\hat{D}$ (see Fig. 8.3(a) for example). Ideally, we want to obtain a design vector $\hat{D}$ with a desirable profile (for example, with the drag rise occurring at a very high Mach number).

Suppose that a "perfect" airfoil $\bar{D}$ could be obtained by some magic process, which might be either theoretical or experimental. Let us look at the unknown process from a reverse engineering point of view. That is, even though we have no idea about how the airfoil was designed, we want to construct an optimization scheme that produces an airfoil with a similar or better profile.

Assume that the profile curve is generated by computing the drag coefficients at Mach numbers $M_1, M_2, \ldots, M_r$. Let $1/\rho_i$ be the value of the drag coefficient of the "perfect" airfoil at Mach number M_i with angle of attack $\bar{\alpha}_i$, which is chosen so that the lift coefficient is equal to c_l^*. Then the global optimal value of (5.3) is no more than 1. If $(\hat{D}, \hat{\alpha}_1, \ldots, \hat{\alpha}_r)$ is a global optimal solution of (5.3), then

$$c_d(\hat{D}, M_i, \hat{\alpha}_i) \le \frac{1}{\rho_i} = c_d(\bar{D}, M_i, \bar{\alpha}_i) \quad \text{for } 1 \le i \le r.$$

Therefore, with the given analysis (*i.e.*, using the profile of the drag coefficient at $M_1, \ldots, M_r$), $\hat{D}$ is no worse than the "perfect" design vector $\bar{D}$.

This shows that for any given profile analysis method, it is possible to recover or improve a desirable airfoil by solving (5.3), if the weights are appropriately chosen. In this sense, (5.3) provides a very flexible tool for airfoil optimization. By experimenting with various choices of the weights ρ_i, a robust airfoil may be obtained (if such an airfoil exists). However, instead of guessing the appropriate weights in (5.3), we can adaptively adjust the weights during the optimization iterations. This leads to the profile optimization method.

Before presenting the profile optimization method, we give a linear programming formulation of a trust-region method for solving (5.3). Using linearization of the nonlinear functions in the mathematically equivalent formulation (5.6) at the current iteration point $(D^k, \alpha_{1,k}, \ldots, \alpha_{r,k})$, we get the following trust-region subproblem for (5.6):

$$\begin{aligned}
(6.1) \qquad & \min_{\Delta D, \Delta\alpha_i, \gamma} \ \gamma \qquad \text{subject to} \\
& -\sigma_i\delta \le \Delta D_i \le \sigma_i\delta \quad \text{for } 1 \le i \le m, \\
& -\alpha_{i,k} \le \Delta\alpha_i \le \alpha_{\max} - \alpha_{i,k} \quad \text{for } 1 \le i \le r, \\
& c_l(D^k, \alpha_{i,k}, M_i) + \left\langle \frac{\partial c_l}{\partial D}(D^k, \alpha_{i,k}, M_i), \Delta D \right\rangle + \frac{\partial c_l}{\partial \alpha}(D^k, \alpha_{i,k}, M_i)\Delta\alpha_i \ge c_l^* \quad \text{for } 1 \le i \le r,
\end{aligned}$$

$$c_d(D^k,\alpha_{i,k},M_i)+\left\langle\frac{\partial c_d}{\partial D}(D^k,\alpha_{i,k},M_i),\Delta D\right\rangle+\frac{\partial c_d}{\partial\alpha}(D^k,\alpha_{i,k},M_i)\Delta\alpha_i\le\frac{\gamma}{\rho_i}\quad\text{for } 1\le i\le r,$$

where ΔD and $\Delta\alpha_i$ are the increments for D and α_i, respectively, σ_i and δ are positive constants that define the trust region for ΔD, and $\alpha_{\max}$ is the maximum angle of attack allowed.

ALGORITHM 6.1. **(Profile Optimization Method)** *Suppose that D^0 is a given design vector and $M_1,M_2,\ldots,M_r$ are the design points. Let $0<\eta<1$ be the predicted percentage reduction rate of the drag for the trust-region method and let $0<\epsilon<1$ be a parameter for termination of the algorithm. Then construct a new design vector as follows.*

(6.1.0) Initialize the angles of attack: *Find $\alpha_{1,0},\alpha_{2,0},\ldots,\alpha_{r,0}$ such that $c_l(D^0,\alpha_{i,0},M_i)=c_l^*$ for $1\le i\le r$; and let $k=0$.*

(6.1.1) Adjust weights: *Let*

$$\rho_i:=\frac{1}{c_d(D^k,\alpha_{i,k},M_i)}\quad\text{for } 1\le i\le r.$$

(6.1.2) Check early termination: *If the zero vector is an optimal solution of (6.1), then output D^k as an optimal solution and terminate the algorithm.*

(6.1.3) Find a trust region for the predicted percentage reduction of the drag: *Find $\delta>0$ such that the optimal objective function value of (6.1) is $(1-\eta)$.*

(6.1.4) Solve the trust-region subproblem: *Find the least norm solution $(\Delta D^k,\Delta\alpha_{1,k},\ldots,\Delta\alpha_{r,k})$ of (6.1).*

(6.1.5) Generate the new iterate: *Let $\alpha_{i,k+1}=\alpha_{i,k}+\Delta\alpha_{i,k}$ for $1\le i\le r$, and $D^{k+1}=D^k+\Delta D^k$.*

(6.1.6) Check heuristic termination conditions: *If*

$$\max_{1\le i\le r}\ \rho_i\,c_d(D^{k+1},\alpha_{i,k+1},M_i)>1\quad\text{and}\tag{6.2}$$

$$\epsilon_k<\epsilon\sum_{i=1}^{r}\ c_d(D^k,\alpha_{i,k},M_i),\tag{6.3}$$

where ϵ_k is the accumulative reduction of the drag at the design points defined by

$$\epsilon_k:=\sum_{i=1}^{r}\ \Big(c_d(D^k,\alpha_{i,k},M_i)-c_d(D^{k+1},\alpha_{i,k+1},M_i)\Big),$$

then output D^k as an optimal solution and terminate the algorithm.

(6.1.7) Start a new iteration: *Update $k:=k+1$ and go back to* **(6.1.1)**.

REMARK 2. The adaptive adjustment of weights makes it possible to achieve a consistent drag reduction over the Mach range $[M_{\min},M_{\max}]$. Note that moving along the descent direction ΔD^k gives a simultaneous drag reduction at all design points $M_1,\ldots,M_r$ (at least if a small step is used). If the selected design points are a "fair representation" of all Mach numbers in $[M_{\min},M_{\max}]$, then moving along ΔD^k will not induce any hidden rise of the drag at some unselected Mach number. This leads to a robust optimization method that achieves a consistent drag reduction over the given Mach range from iteration to iteration (see the profiles of the drag coefficient for iterates generated by the profile optimization method in Fig. 8.4(a)).

In contrast, without adaptive weight adjustments, it is necessary to use at least $(m+1)$ design points, where m is the number of free-design variables, in order to avoid point-optimization at selected design points (cf. Sections 3 and 5).

Besides the adaptive adjustment of weights, there are three features of the profile optimization method that are not standard for a trust-region method.

(a) First, the size of the trust region is modified to achieve a given predicted percentage reduction rate η of the drag. This approach is employed to avoid the following two problems, which are incurred in practical applications of a standard trust-region method: (i) an appropriate size of the trust region is unknown without prior experiences, and (ii) if each iteration takes a long time to complete, then rejecting a new iterate might be hard to accept by a designer.

For airfoil shape optimization, if the percentage reduction of c_d is too small, then the reduction could be a consequence of numerical errors and is less reliable. If the predicted percentage reduction of c_d is too large, then the size of the trust region is also relatively large and the predicted reduction can not be trusted. By choosing an appropriate percentage reduction rate for each iteration, these two problems might be avoided.

Note that the choice of the predicted percentage reduction rate depends on a user's knowledge of the accuracy of the simulation analysis tool involved, the time allowed to generate a new design, and the expected overall performance improvement. For example, if the relative numerical error for computing c_d is 0.5%, each iteration takes 3 hours, a designer has 1 day to generate a new design, and the expected improvement is 8%, then the designer could set $\eta = 1\%$ so that each iteration will produce some meaningful improvement of the design and the final design might have about 8% improvement over the original design after 1 day (or 8 iterations).

(b) The second non-standard feature is that we use the least norm solution of the linear programming problem (6.1), which can be obtained by solving a quadratic perturbation of (6.1) [14]. Our implicit assumption here is that the original airfoil is reasonable. Therefore, we do not want a new airfoil to deviate too much from the original one if unnecessary. By using the least norm solution of (6.1), we intend to select a new airfoil closest to the original one while achieving a predetermined amount of improvement in performance.

(c) The third non-standard feature of the profile optimization method is the termination criterion, which is based on design heuristics. The basic idea is that if a good descent direction for drag reduction at all design points does not work (*i.e.*, (6.2) holds) and the overall percentage reduction is insignificant (*i.e.*, (6.3) holds), then there is no need to continue the optimization process. ■

7. Implementation of Profile Optimization Method. The main implementation issues related to the profile optimization method are the following:

- parallel processing of the function and gradient evaluations for r Mach numbers,
- finding initial values of the angles of attack for r Mach numbers,
- finding the size η for the trust region in an iteration,
- computing the least norm solution of (6.1).

In the next four subsections, we discuss these four implementation issues.

7.1. Parallel Processing of Function and Gradient Evaluations. Approximate values of c_l (or c_d) can be obtained with one flow analysis (*i.e.*, solving the PDE once) and the gradient of c_l (or c_d) can be obtained with one sensitivity analysis (*i.e.*, solving one adjoint equation). The function values and the gradients of c_d (or c_l) at different Mach numbers can be calculated independently. Therefore, if $2r$ processors are available in a parallel computing environment, the walltime for function and gradient evaluations in each iteration of the profile optimization is one flow analysis and one sensitivity analysis.

It is desirable to have a parallel wrapper code for the profile optimization method to run several instances of a CFD code without requiring any modifications to the code. For this purpose, we store multiple copies of an executable CFD code (FUN2D [15] for the current implementation) under different directories (called hosting directories) in a file server. Each copy of the CFD code communicates with the profile optimization

method by writing its output into its own hosting directory and the profile optimization method collects all function values and gradients from the hosting directories. The architecture of a parallel computing environment (such as shared memory or distributed memory) has no impact on the implementation since there is no communication between any two different processors.

The current parallel wrapper code is based on a shell-level parallel computing protocol "pbsdsh" on Coral [13] at ICASE. Coral is a Beowulf-class cluster of 96 Intel Pentium processors with 3 additional file servers. To get the function values and the gradients for c_l and c_d at r different Mach numbers simultaneously, we just create $2r$ hosting directories. For $1 \le i \le r$, in the i-th directory, we use FUN2D to compute the function value and the gradient of c_l for the i-th Mach number M_i, and in the $(r+i)$-th directory, we use FUN2D to compute the function value and the gradient of c_d for the i-th Mach number M_i. Therefore, with $2r$ processors, we get all function values and gradients for c_l and c_d in the walltime for one flow analysis and one sensitivity analysis.

Note that on Coral, the Unix command "pbsdsh gradfun" makes the same executable code "gradfun" run on different nodes of Coral in parallel. Here is a brief description of how to write a C-code that generates an executable code "gradfun" for parallel processing of function and gradient evaluations on Coral:

- Create a shell script called "get_id", whose content is

  echo $PBD_NODENUM

  The output of "get_id" is the value of the environment variable $PBD_NODENUM, which is $(i-1)$ if "get_id" is executed on the i-th node.
- Write a C-code that does the following:
  - use "gethostname(HOST_NAME)" to get the host name (such as n013) of the node running the C-code;
  - execute a system command

    "get_id > HOST_NAME"

    in the C-code that gets the $PBD_NODENUM environment variable and stores it in a file whose name is the content of the string HOST_NAME;
  - read the index number id in the file whose name is the content of HOST_NAME;
  - go to the id-th hosting directory; and
  - execute FUN2D to get the function value and gradient for c_l (or c_d) at M_i if and only if $i = id+1$ (or $r + i = id + 1$) for $1 \le i \le r$.

7.2. Finding Initial Values of the Angles of Attack. For a fixed airfoil shape and a fixed Mach number, the lift is almost a linear function of the angle of attack [10, p. 99]. Therefore, we use the following search method based on a linear interpolation to find an angle of attack for which the corresponding lift is the given target value.

ALGORITHM 7.1. *Choose an error tolerance $\epsilon > 0$ and an initial adjustment rate $0 < \kappa < 1$. Let α_0 be an initial guess of the angle of attack for a given Mach number M and a given geometric design vector D. Then the following procedure will find a value of α such that $|c_l(D, M, \alpha) - c_l^*| < \epsilon$.*

(7.1.0) Initialize the range variables α_1 and α_2: *If $c_l(D, M, \alpha_0) > c_l^*$, then $\alpha_1 = (1-\kappa)\alpha_0$ and $\alpha_2 = \alpha_0$; otherwise, $\alpha_1 = \alpha_0$ and $\alpha_2 = (1+\kappa)\alpha_0$.*

(7.1.1) Use linear interpolation to predict α: *Let*

$$\alpha = \alpha_1 + \frac{c_l^* - c_l(D, M, \alpha_1)}{c_l(D, M, \alpha_2) - c_l(D, M, \alpha_1)}(\alpha_2 - \alpha_1).$$

(7.1.2) Check the termination condition: *If*

$$|c_l(D, M, \alpha) - c_l^*| < \epsilon\, c_l^*,$$

then output α *and terminate the algorithm.*

(7.1.3) Update α_1 **and** α_2**:**

- *if* $\alpha < \alpha_1$*, then* $\alpha_2 = \alpha_1$ *and* $\alpha_1 = \alpha$*;*
- *if* $\alpha > \alpha_2$*, then* $\alpha_1 = \alpha_2$ *and* $\alpha_2 = \alpha$*;*
- *if* $\alpha_1 < \alpha < \alpha_2$ *and* $c_l(D, M, \alpha) < c_l^*$*, then* $\alpha_1 = \alpha$*;*
- *if* $\alpha_1 < \alpha < \alpha_2$ *and* $c_l(D, M, \alpha) > c_l^*$*, then* $\alpha_2 = \alpha$*.*

(7.1.4) Start a new iteration: *Go back to* **(7.1.1)**.

7.3. Finding the Size of the Trust Region. It is well-known that the optimal objective function value of (7.1) is a non-increasing and piecewise affine function of δ. Therefore, we use the following search method based on a linear interpolation to find δ.

ALGORITHM 7.2. *Let* $\gamma(\delta)$ *be the optimal function value of (7.1) for any given* δ*. Choose an error tolerance* $\epsilon > 0$ *and an initial adjustment rate* $0 < \kappa < 1$*. If the zero vector is not an optimal solution of (7.1) for* $\delta = \delta_0$ *with* $0 < \delta_0 < 1$*, then for any* $\epsilon > 0$*, the following procedure will find* $0 < \delta < 1$ *such that* $|\gamma(\delta) - (1 - \eta)| < \epsilon$*.*

(7.2.0) Initialize the range variables δ_1 **and** δ_2**:** *If* $\gamma(\delta_0) > 1 - \eta$*, then* $\delta_1 = \delta_0$ *and* $\delta_2 = (1 + \kappa)\delta_0$*; otherwise,* $\delta_1 = (1 - \kappa)\delta_0$ *and* $\delta_2 = \delta_0$*.*

(7.2.1) Use linear interpolation to predict δ**:** *Let*

$$\delta = \delta_1 + \frac{(1 - \eta) - \gamma(\delta_1)}{\gamma(\delta_2) - \gamma(\delta_1)}(\delta_2 - \delta_1).$$

(7.2.2) Check the termination condition: *If*

$$|\gamma(\delta) - (1 - \eta)| < \epsilon,$$

then output δ *and terminate the algorithm.*

(7.2.3) Update δ_1 **and** δ_2**:**

- *if* $\delta < \delta_1$*, then* $\delta_2 = \delta_1$ *and* $\delta_1 = \delta$*;*
- *if* $\delta > \delta_2$*, then* $\delta_1 = \delta_2$ *and* $\delta_2 = \delta$*;*
- *if* $\delta_1 < \delta < \delta_2$ *and* $\gamma(\delta) > (1 - \eta)$*, then* $\delta_1 = \delta$*;*
- *if* $\delta_1 < \delta < \delta_2$ *and* $\gamma(\delta) < (1 - \eta)$*, then* $\delta_2 = \delta$*.*

(7.2.4) Start a new iteration: *Go back to* **(7.2.1)**.

REMARK 3. In general, we use the value of δ in the previous iteration of the profile optimization method as δ_0. With this choice of δ, $\kappa = 0.02$, and $\epsilon = 0.01 \cdot \eta$, we usually solve about 4 linear programming subproblems to find δ in our numerical simulation runs. ■

7.4. Least Norm Solution of LP Subproblem. By using slack variables u_i and v_i, we can convert (6.1) into a linear programming problem with equality constraints and simple bound constraints:

$$(7.1) \quad \min_{\Delta D, \Delta\alpha_i, \gamma, u_i, v_i} \gamma \qquad \text{subject to}$$

$$0 \le u_i \le c_l^*, \quad 0 \le v_i \le 1, \quad \text{for } 1 \le i \le r,$$

$$-\alpha_{i,k} \le \Delta\alpha_i \le \alpha_{\max} - \alpha_{i,k} \quad \text{for } 1 \le i \le r,$$

$$-\sigma_i\delta \le \Delta D_i \le \sigma_i\delta \quad \text{for } 1 \le i \le m,$$

$$c_l(D^k, \alpha_{i,k}, M_i) + \left\langle \frac{\partial c_l}{\partial D}(D^k, \alpha_{i,k}, M_i), \Delta D \right\rangle + \frac{\partial c_l}{\partial \alpha}(D^k, \alpha_{i,k}, M_i)\Delta\alpha_i - u_i = c_l^* \quad \text{for } 1 \le i \le r,$$

$$c_d(D^k, \alpha_{i,k}, M_i) + \left\langle \frac{\partial c_d}{\partial D}(D^k, \alpha_{i,k}, M_i), \Delta D \right\rangle + \frac{\partial c_d}{\partial \alpha}(D^k, \alpha_{i,k}, M_i)\Delta\alpha_i + v_i = \frac{\gamma}{\rho_i} \quad \text{for } 1 \le i \le r.$$

The least norm solution of (7.1) can be found by solving a strictly convex quadratic programming problem [14] that is derived by replacing the objective function in (7.1) by

$$\gamma + \frac{s}{2}\left(\gamma^2 + \sum_{i=1}^{r}\left[u_i^2 + v_i^2 + (\Delta\alpha_i)^2\right] + \sum_{i=1}^{m}(\Delta D_i)^2\right), \tag{7.2}$$

where s ($= 10^{-8}$ in our implementation) is a small positive scalar.

Note that minimizing the objective function in (7.2) will force all v_i's to be more or less the same. As a consequence, it produces a descent direction that gives an almost identical relative reduction of the drag for all the design points $M_1, \ldots, M_r$. We believe that this balanced reduction of drag will prevent point-optimization at the sampled design points. Furthermore, the least norm solution generates a new design vector that stays as close to the original design vector as possible while achieving the predetermined percentage reduction of the drag. This is desirable in a practical design process, since if a new design deviates too far away from the original airfoil, then designers might have less confidence in the predicted performance of the new airfoil.

8. Numerical Simulation Results. We use the NACA-0012 airfoil as the initial point for all optimization methods discussed in this section. A periodic spline representation of the NACA-0012 with 23 control points is used to get an initial parametric model of the airfoil:

$$x = \sum_{i=0}^{22} a_i\, p_i(t), \quad y = \sum_{i=0}^{22} b_i\, q_i(t), \quad \text{for } 0 \le t \le 1,$$

where p_i and q_i are spline basis functions. The shape of the airfoil can be modified by changing the values of the b_i's. The tip and the tail of the airfoil are fixed during the optimization (*i.e.*, b_0, b_{11}, and b_{22} are fixed) so that the chord length remains constant. As a consequence, we have 20 free-design variables $(b_1, \ldots, b_{10}, b_{12}, \ldots, b_{21})$ in the airfoil shape optimization.

The CFD code FUN2D is used to compute the function values and gradients of the lift and drag [15]. Some technical details of the flow solver (for function evaluations) and the adjoint equation solver (for gradient evaluations) in FUN2D can be found in [1] and [16], respectively.

We use Euler flow to demonstrate the usefulness of the profile optimization method as a robust airfoil optimization tool. The main reason to choose Euler flow simulation analysis is that we can complete our preliminary evaluation of the profile optimization method in a reasonable amount of time.

Elliott and Peraire [6, Section 1] stated that almost any drag minimization exercise based on the Euler equations and applied to modern supercritical wings in cruise condition is doomed to failure. However, they also concluded that Euler-based optimization can be useful as a first step in the design process [7, Abstract].

For a given initial grid, FUN2D generates numerical approximations to the lift/drag coefficients and their derivatives. We use 10^{-9} as the error tolerance for the flow solver in FUN2D to find an approximate solution of the nonlinear system derived from discretization of the Euler equation. The number of time steps used in solving the discrete adjoint equation is 10000 and the spring analogy method in FUN2D is used for grid movement [16].

The initial grid for FUN2D used in this paper has 124 points around the airfoil and 32 points at the far field (at a distance of 50 chord lengths). The grid has 3060 nodes, 9025 faces, and 6120 elements. Fig. 8.1 shows the initial grid around the NACA-0012. See [11] for a comparison of the numerical approximations of the lift and drag using this grid and other grids.

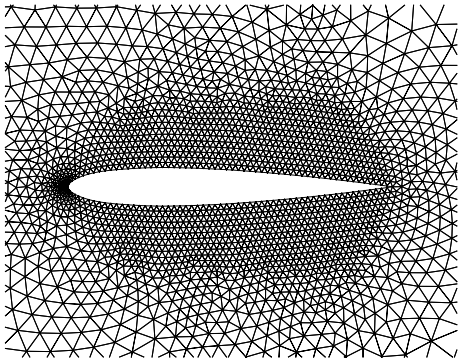


FIG. 8.1. *The grid used for solving the Euler equation by FUN2D*

All simulation results are obtained by parallel processing of function and gradient evaluations on Coral (cf. Subsection 7.1). For the grid we use, it takes about 120 seconds to solve a flow problem and about 180 seconds to solve an adjoint equation. Therefore, with $2r$ processors, each iteration of the profile optimization method takes about 5 minutes, no matter what the number of design points is.

We compute the least norm solution of (7.1) by the quadratic programming solver QLD, which was first developed by Powell [18] and then modified by Schittkowski [19].

Our numerical simulations are designed to examine the impact of the number of design points and the target lift value on the profile optimization method. Also, we shall compare the simulation results obtained by the profile optimization method with the multipoint optimization method and the expected value optimization method. Therefore, we consider the following cases for numerical simulation:

1. the profile optimization with 4 design points equally spaced in the Mach range $[0.7, 0.8]$, $c_l^* = 0.4$, a fixed percentage reduction rate $\eta = 5\%$, and a termination parameter $\epsilon = 0.3\%$;
2. the profile optimization with 3 design points equally spaced in the Mach range $[0.7, 0.8]$, $c_l^* = 0.4$, a fixed percentage reduction rate $\eta = 5\%$, and a termination parameter $\epsilon = 0.3\%$;
3. the profile optimization with 8 design points equally spaced in the Mach range $[0.7, 0.8]$, $c_l^* = 0.4$, a fixed percentage reduction rate $\eta = 5\%$, and a termination parameter $\epsilon = 0.3\%$;
4. the profile optimization with 4 design points equally spaced in the Mach range $[0.7, 0.8]$, $c_l^* = 0.2$, a fixed percentage reduction rate $\eta = 5\%$, and a termination parameter $\epsilon = 0.3\%$;
5. the multipoint airfoil optimization with 4 design points equally spaced in the Mach range $[0.7, 0.8]$, $w_1 = w_4 = 1/6$, $w_2 = w_3 = 1/3$, and $c_l^* = 0.4$;
6. the expected value airfoil optimization with adaptive changes of w_i corresponding to 4 randomized integration points, and $c_l^* = 0.4$.

The first case is the benchmark case. The next two cases are included to examine the impact of the number of design points on the profile optimization. We use Case 4 to examine the impact of the target lift value on the profile optimization. The last two cases are for comparison of the profile optimization with

other optimization methods. The profiles of the drag coefficient for a given c_l^* are plotted by using 24 equally spaced Mach numbers in $[0.7, 0.8]$.

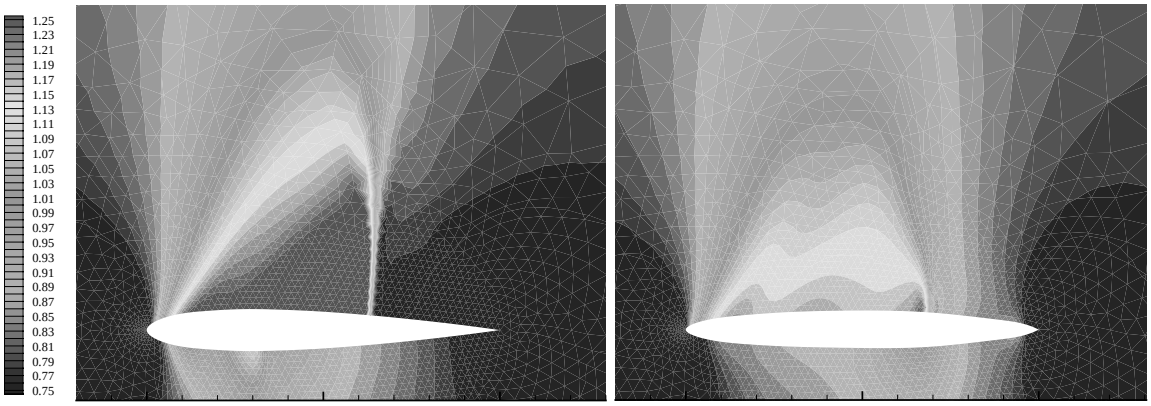
8.1. Impact of the Number of Design Points. The profile optimization method terminates after 69, 67, and 57 iterations for 3, 4, and 8 design points, respectively. The number of design points has no significant impact on the optimal airfoils generated by the profile optimization method. All optimal airfoils have similar shapes (cf. Table 8.1) and only a marginal drag reduction is achieved by adding more iterations (cf. Table 8.2).

TABLE 8.1
The relative differences (in percentage) of the design vectors

	NACA-0012	3 Points	4 Points	8 Points
NACA-0012	–	24.5	25.7	23.8
3 Points	24.7	–	1.7	1.2
4 Points	25.9	1.8	–	2.6
8 Points	23.9	1.2	2.6	–

Each number in Table 8.1 denotes the relative difference of two design vectors: $\|\mathbf{D}_r - \mathbf{D}_c\|/\|\mathbf{D}_r\|$, where $\mathbf{D}_r$ (or $\mathbf{D}_c$) is either the NACA-0012 or the optimal design vector obtained by the profile optimization method with the number of design points listed in the corresponding row (or column).

A typical optimal airfoil is given in Fig. 8.2(b). This Mach wave plot illustrates how one strong shock wave for the NACA-0012 is reduced to several weaker shock waves for the optimal airfoil.



(a) NACA-0012 (b) Profile Optimization

FIG. 8.2. *The Mach waves (for the Mach number 0.796) around the NACA-0012 and an optimal airfoil generated by the profile optimization method*

With 4 design points, the profile optimization method achieves consistent drag reduction over the Mach range $[0.7, 0.8]$ (cf. Fig. 8.3(a) and 8.4(a)). There is no random distortion of airfoil shapes during the optimization process (cf. Fig. 8.3(b) and 8.4(b)).

It is worth pointing out that the iterates generated by the profile optimization method are also independent of the number of design points. For example, the 56-th iterate generated by the profile optimization method for 4 design points differs from the 56-th iterate corresponding to 3 (or 8) design points by 0.35% (or 1.2%). The profiles of the drag coefficient for optimal airfoils generated by the profile optimization with

3 and 4 design points, respectively, are almost identical. The profile of the drag coefficient for the optimal airfoil corresponding to 8 design points is very similar to the drag profile of the 57-th iterate generated by the profile optimization with 4 design points.

8.2. Impact of the Target Lift Value. The target lift value c_l^* has a quite significant impact on the optimal airfoil generated by the profile optimization method.

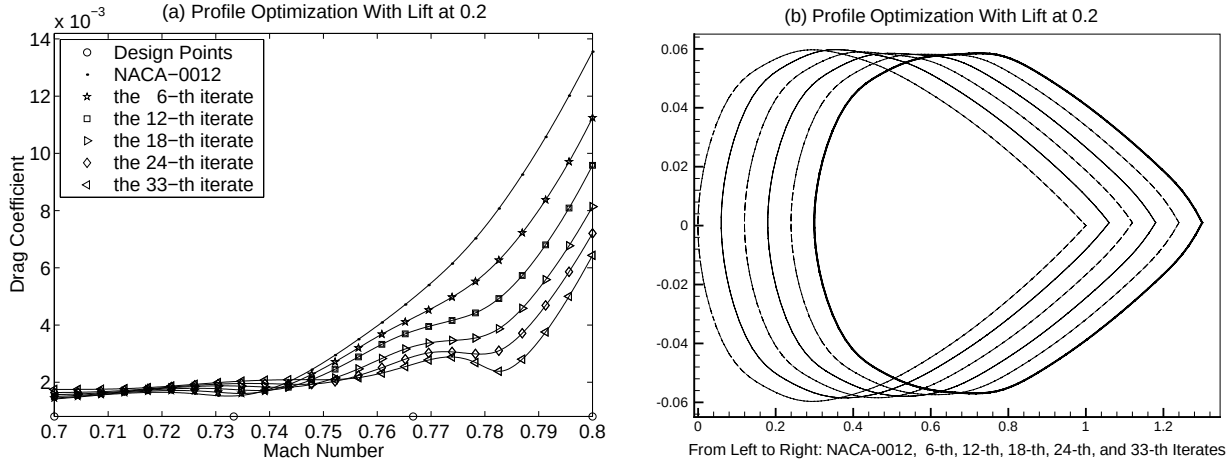


FIG. 8.3. *The two figures illustrate the behaviors of the iterates generated by the profile optimization method, with $c_l^* = 0.2$ and 4 equally spaced design points. Fig. 8.3(a) shows profiles of the drag coefficient versus the Mach number for the iterates and Fig. 8.3(b) shows the shapes of airfoils corresponding to the iterates. (Note. The airfoils are shifted to improve the visibility.)*

With $c_l^* = 0.2$, the NACA-0012 is almost optimal for low Mach numbers. Therefore, we can not simultaneously reduce the drag over the Mach range $[0.7, 0.8]$. The profile optimization method tries to keep the drag as low as possible for Mach numbers between 0.7 and 0.75, while reducing the drag significantly for Mach numbers between 0.75 and 0.8 (cf. Fig. 8.3(a)).

Note that the drag bucket (*i.e.*, a drop of the drag before its dramatic rise) occurs near $M = 0.73$ for the NACA-0012 and the drag bucket for the optimal airfoil occurs near $M = 0.78$. Such a delay of the drag rise is desirable. The only compromise made by the optimal airfoil to the NACA-0012 is a small increase of the drag around the original drag bucket for the NACA-0012, which is quite reasonable.

For the higher target lift $c_l^* = 0.4$, the optimal airfoil shape (shown in Fig. 8.4(b)) deviates more from the NACA-0012. Also, the 33rd iterate generated by the profile optimization method for $c_l^* = 0.4$ differs from the 33rd iterate for $c_l^* = 0.2$ by 10%.

Note that the shape variations of the airfoils corresponding to iterates generated by the profile optimization method follow a similar trend no matter what the target lift is: the aft end thickens while the front section narrows. This pushes the shock location towards the aft region of the airfoil.

A similar airfoil with thin front and fat tail was obtained by Elliott and Peraire [6, Figure 34] in a totally different context, where they used two thickness design variables and one camber design variable for a lift-constrained drag optimization with 2-D separated viscous flow and an additional area constraint. The initial airfoil used by them is also the NACA-0012. Their explanation of the merit of an airfoil with thinner front and fatter tail over the NACA-0012 is that the thickness distribution has been redistributed such that the maximum is further aft (*i.e.*, closer to the tail), like the early natural laminar flow airfoils, this delays the start of the adverse pressure gradient to aft of that maximum thickness point.

8.3. Impact of the Optimization Strategy. From the analysis given in Sections 3 and 5, we know that if the design points and weights are fixed as in either (5.1) or (5.3), then a severe degradation in the off-design performance is likely when the number of design points (*i.e.*, 4) is much smaller than the number of free-design variables (*i.e.*, 20). Fig. 8.5(a) clearly reveals the point-optimization features of the optimal airfoil generated by the multipoint optimization method. Fig. 8.4(c) suggests that the point-optimization behavior does not occur until the end of the optimization iterations, since the drag curve for the 25-th iterate does not have any drag trough.

The expected value optimization method [12] adds a random perturbation to the integration points from iteration to iteration and adjusts the weights accordingly so that a severe degradation in the off-design performance can be avoided. Fig. 8.4(e) shows that this technique allows the optimizer to reduce the drag while avoiding point-optimization.

The profile optimization avoids the problem of point-optimization at the design points (cf. Fig. 8.3(a) and 8.4(a)) by using a smart descent direction to achieve consistent drag reduction over the whole Mach range.

TABLE 8.2
Relative drag reduction rates (in percentage)

	Max	Min	Average
Profile Optimization (3 Points)	82.9	5.1	70.2
Profile Optimization (4 Points)	82.7	1.4	69.7
Profile Optimization (8 Points)	82.6	4.1	67.9
Multipoint Optimization (4 Points)	89.4	46.2	81.1
Expected Value Optimization (4 Points)	89.4	49.6	82.6

Table 8.2 gives an overall sense of how each optimization method reduces the drag coefficient with $c_l^* = 0.4$. Let $M_1, M_2, \ldots, M_{24}$ be 24 equally spaced points in $[0.7, 0.8]$. For each M_i, let $c_{d,i}$ (or $\hat{c}_{d,i}$) be the drag coefficient of the NACA-0012 (or an optimal airfoil) with the lift coefficient fixed at 0.4. Then the relative reduction of the drag at each Mach number M_i is $(c_{d,i} - \hat{c}_{d,i})/c_{d,i}$. The numbers in "Max", "Min", and "Average" columns of Table 8.2 correspond to

$$\max_{1\le i\le 24} \frac{c_{d,i} - \hat{c}_{d,i}}{c_{d,i}}, \quad \min_{1\le i\le 24} \frac{c_{d,i} - \hat{c}_{d,i}}{c_{d,i}}, \text{ and}$$

$$\left(\sum_{i=1}^{24} c_{d,i} - \sum_{i=1}^{24} \hat{c}_{d,i}\right) \Big/ \left(\sum_{i=1}^{24} c_{d,i}\right), \text{ respectively.}$$

Note also that the expected value optimization method (as well as the multipoint optimization method) resulted in an optimal airfoil with a drag curve that is consistently lower over the Mach range than the profile optimization method (cf. Fig. 8.5(a)). It seems that it should be possible for the profile optimization method to find a descent direction for a consistent drag reduction over the whole Mach range $[0.7, 0.8]$, by moving toward the optimal solution given by either the multipoint optimization method or the expected value optimization method. However, due to nonlinearity of the drag coefficient (with respect to the design variables), the profile optimization method can not find such a descent direction by using the local derivative information only.

9. Conclusion. In this paper, we introduce a new robust optimization scheme called the profile optimization method and use airfoil optimization under uncertain flight conditions as a case study to evaluate

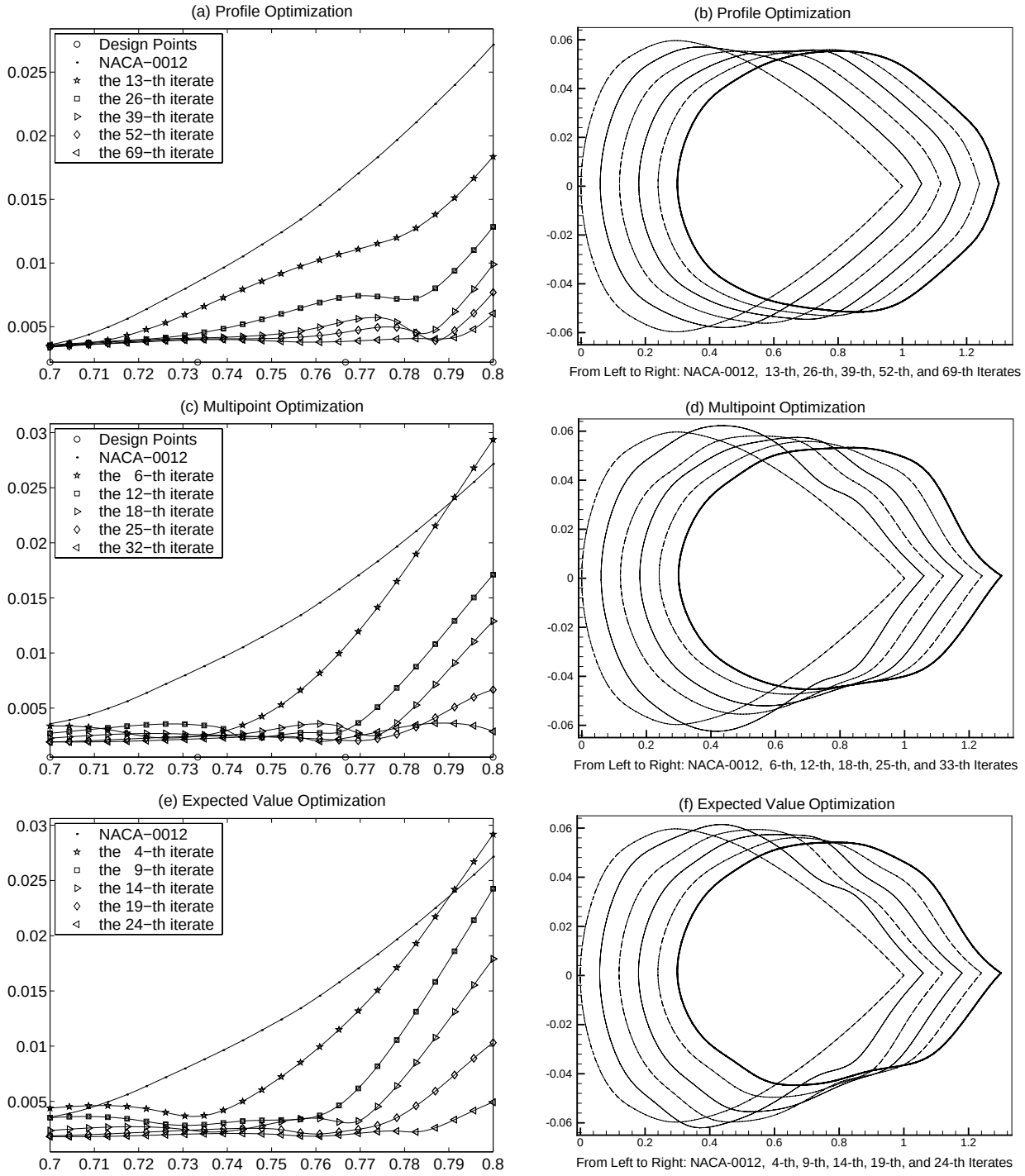


FIG. 8.4. *These figures show the behaviors of iterates generated by various optimization methods with* $c_l^* = 0.4$ *and 4 design points.*

this method. We used an Euler-based CFD code, which does not include viscous effects, to test the profile optimization method. Because of this lack of fidelity, the generated airfoils may be somewhat unrealistic.

The profile optimization method adaptively adjusts the weights in a minimax optimization formulation to find a drag reduction direction for all design conditions, which leads to a consistent drag reduction over

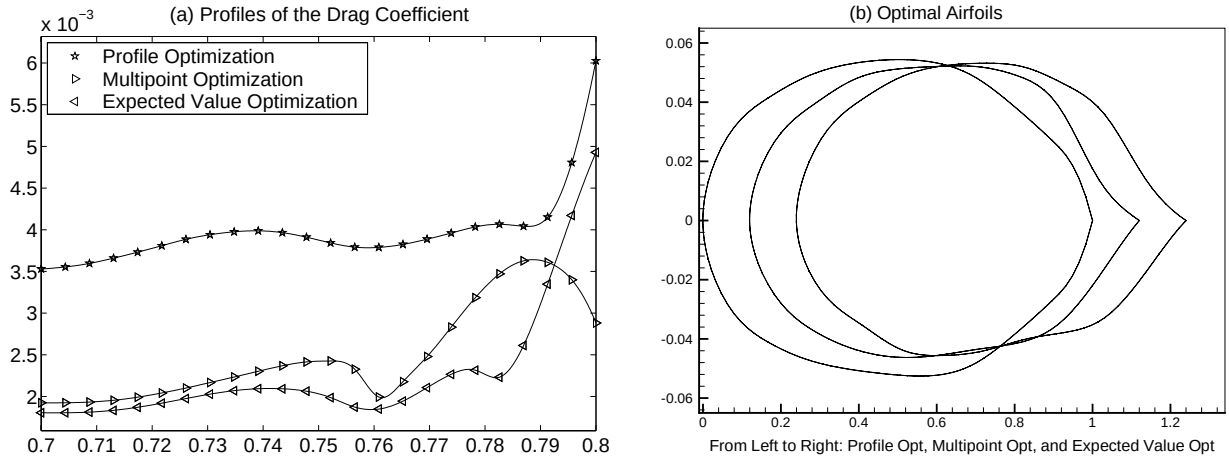


FIG. 8.5. *Fig. 8.5(a) compares the profiles of the drag coefficient for various optimal airfoils and Fig. 8.5(b) compares the optimal airfoil shapes.*

a given range of Mach numbers from iteration to iteration.

Without adaptive adjustment of weights, we prove that it is necessary to use at least $(m+1)$ design points, where m is the number of free-design variables, in order to avoid point-optimization at selected design points.

Our numerical results demonstrate that the profile optimization method is not sensitive to the number of selected design points. With 20 free geometric design variables and as little as 4 design points, the optimal airfoil generated by the profile optimization method is smooth and has no degradation in the off-design performance.

The profile optimization method can be easily modified to solve other optimization problems under uncertainty by replacing c_d with another performance measurement function and M with other uncertain parameters.

The profile optimization method also has the potential of becoming a practical design tool for optimization under uncertainty. The use of a small number of sampled design points from the range of uncertain parameters makes the profile optimization method computationally affordable. The consistent performance improvements over the range of uncertain parameters from iteration to iteration allows a designer to stop the iterative process at any time with an improved new design.

Acknowledgement. We are grateful to Perry A. Newman at the MDO Branch of NASA Langley Research Center and Manuel D. Salas at ICASE for their insightful comments on aerodynamics related to airfoil designs, and to Eric Nielsen at the Aerodynamic and Acoustic Methods Branch of NASA Langley Research Center for his expertise of FUN2D. We also extend our thanks to Michael Lewis for suggesting airfoil optimization as a case study of robust optimization methods.

REFERENCES

[1] W. ANDERSON AND D. BONHAUS, An implicit upwind algorithm for computing turbulent flows on unstructured grids, Computers and Fluids, 23 (1994), 1–21.

[2] W. Anderson and D. Bonhaus, Airfoil optimization on unstructured grids for turbulent flows, AIAA Journal, 37 (1999), 185–191.

[3] W. Anderson and V. Venkatakrishnan, Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation, AIAA-97-0643, 1997.

[4] A. Ben-Tal and A. Nemirovski, Robust truss topology design via semidefinite programming, SIAM J. Optim., 7 (1997), 991–1016.

[5] M. Drela, Pros and cons of airfoil optimization, in "Frontiers of Computational Fluid Dynamics 1998", edited by D.A. Caughey and M.M. Hafez, World Scientific, 1998.

[6] J. Elliott and J. Peraire, Aerodynamic optimization on unstructured meshes with viscous effects, AIAA-97-1849, 1997.

[7] J. Elliott and J. Peraire, Constrained, multipoint shape optimization for complex 3-D configurations, Aeronautical Journal, 102 (1998), 365–376.

[8] W. Fowlkes and C. Creveling, *Engineering Methods for Robust Product Design Using Taguchi Methods in Technology and Product Development*, Addison-Wesley, Reading, MA, 1995.

[9] G. Gumbert, G. Hou, and P. Newman, Simultaneous aerodynamic analysis and design optimization (SAADO) for a 3-D flexible wing, 39th AIAA Aerospace Science Meeting and Exhibit, Reno, Nevada, AIAA-2001-1107, 2001.

[10] E. Houghton and A. Brock, Aerodynamics for Engineering Students, Edward Arnold (Publishers) LTD., London, 1960.

[11] L. Huyse, Free-form airfoil shape optimization under uncertainty using maximum expected value and second-order second-moment strategies, NASA/CR-2001-211020 or ICASE Report No. 2001-18, 2001.

[12] L. Huyse and R. Lewis, Aerodynamic shape optimization of two-dimensional airfoils under uncertain operating conditions, NASA/CR-2001-210648 or ICASE Report No. 2001-1, 2001.

[13] J. Loncaric, ICASE, http://www.icase.edu/CoralProject.html, 2001.

[14] O. Mangasarian, Normal solutions of linear programs, Math. Prog. Study, 22 (1984), 206–216.

[15] E. Nielsen, NASA LaRC, http://fmad-www.larc.nasa.gov/~nielsen/Fun/fun.html, 2001.

[16] E. Nielsen and W. Anderson, Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations, AIAA-98-4809, 1998.

[17] E. Nielsen and W. Anderson, Recent improvements in aerodynamic design optimization on unstructured meshes, AIAA-2001-0596, 2001.

[18] M. Powell, ZQPCVX, a FORTRAN subroutine for convex programming, Report DAMTP/1983/NA17, University of Cambridge, England, 1983.

[19] K. Schittkowski, QLD – A FORTRAN code for quadratic programming, User's Guide. Mathematisches Institut, Universität Bayreuth, Germany, 1986.